Ri Material Binding

Basics

In Ri, shaders are bound the last active Bxdf. In this case, two difference Bxdfs named "disney1" are bound to two different spheres.

```
AttributeBegin
    Bxdf "PxrDisney" "disney1" "color base color" [0 1 1]
    Translate -1 0 0
        Sphere 1 -1 1 360
AttributeEnd
AttributeBegin
    Bxdf "PxrDisney" "disney1" "color base color" [0 1 1]
    Translate 1 0 0
        Sphere 1 -1 1 360
AttributeEnd
```

Deduplication

Because attribute and transformation hierarchies often don't match material hierarchies, material network may be duplicated in Ri streams. In the above example, the two Bxdfs are identical. RenderMan provides a special "string __materialid" mechanism to identify and re-bind to previously emitted materials. In this case, a single "disney1" material network is emitted but bound to two different spheres. The same "string __materialid" parameter can use used to identify and deduplicate RiBxdf, RiLight, and RiDisplace shading networks.

```
AttributeBegin
    Bxdf "PxrDisney" "disney1" "color base color" [0 1 1] "string __materialid" ["material1"]
AttributeEnd
AttributeBegin
    Bxdf "" "" "string __materialid" ["material1"]
    Translate -1 0 0
    Sphere 1 -1 1 360
AttributeEnd
AttributeBegin
    Bxdf "" "" "string __materialid" ["material1"]
    Translate 1 0 0
    Sphere 1 -1 1 360
AttributeEnd
```

Late Binding

The same mechanism can be used to late bind materials inside Ri procedurals. In this case, a sphere is bound to material "PxrDiffuse1SG".

```
static RtUString const k_materialid("string __materialid");
RtUString materialid("PxrDiffuse1SG");
RtUString nms[] = { k_materialid };
RtPointer vals[] = { &materialid };
ri->BxdfV(Rix::k_empty, Rix::k_empty, 1, nms, vals);
ri->SphereV(1, -1, 1, 360, 0, nullptr, nullptr);
```